EAST Search History

Ref #	Hits	Search Query	DBs	Default Operator	Plurals	Time Stamp
L1	3	heap near2 generat$3 same (virtual adj1 machine)	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 10:58
L2	61	heap near2 generat$3 same (garbage near2 collect$3)	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 10:58
L3	4824484	@ad<="20000812"	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 10:59
L4	26	l2 and l3	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 11:04
L5	10	heap near2 reset same card	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 11:06
L6	0	l5 and l3	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 11:04
L7	6	heap adj12 reset same card	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 11:25
L8	72	card near2 mark$3 and heap	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 11:25
L9	37	l8 and l3	US-PGPUB; USPAT; USOCR; EPO	OR	ON	2006/03/27 11:25

Terms used
heap reset and **garbage collection** and **card marking**

Found **17,713** of **171,143**

Sort results by: relevance
Display results: expanded form
Save results to a Binder
Search Tips
Open results in a new window

Try an Advanced Search
Try this search in The ACM Guide

Results 21 - 40 of 200 Result page: previous 1 **2** 3 4 5 6 7 8 9 10 next
Best 200 shown Relevance scale

21 MC²: high-performance garbage collection for memory-constrained environments

Narendran Sachindran, J. Eliot B. Moss, Emery D. Berger
October 2004 **ACM SIGPLAN Notices , Proceedings of the 19th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '04**, Volume 39 Issue 10
Publisher: ACM Press

Full text available: pdf(503.53 KB) Additional Information: full citation, abstract, references, citings, index terms

Java is becoming an important platform for memory-constrained consumer devices such as PDAs and cellular phones, because it provides safety and portability. Since Java uses garbage collection, efficient garbage collectors that run in constrained memory are essential. Typical collection techniques used on these devices are mark-sweep and mark-compact. Mark-sweep collectors can provide good throughput and pause times but suffer from fragmentation. Mark-compact collectors prevent fragmentation, ...

Keywords: copying collector, generational collector, java, mark-compact, mark-copy, mark-sweep, memory-constrained copying

22 Mostly concurrent garbage collection revisited

Katherine Barabash, Yoav Ossia, Erez Petrank
October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11
Publisher: ACM Press

Full text available: pdf(279.42 KB) Additional Information: full citation, abstract, references, citings, index terms

The *mostly concurrent garbage collection* was presented in the seminal paper of Boehm et al. With the deployment of Java as a portable, secure and concurrent programming language, the mostly concurrent garbage collector turned out to be an excellent solution for Java's garbage collection task. The use of this collector is reported for several modern production Java Virtual Machines and it has been investigated further in academia.In this paper, we present a modification of the mostly concu ...

Keywords: JVM, Java, concurrent garbage collection, garbage collection, incremental garbage collection

23 Understanding memory allocation of scheme programs

Manuel Serrano, Hans-J. Boehm

September 2000 **ACM SIGPLAN Notices , Proceedings of the fifth ACM SIGPLAN international conference on Functional programming ICFP '00**, Volume 35 Issue 9

Publisher: ACM Press

Full text available: pdf(821.49 KB) Additional Information: full citation, abstract, references, citings, index terms

Memory is the performance bottleneck of modern architectures. Keeping memory consumption as low as possible enables fast and unobtrusive applications. But it is not easy to estimate the memory use of programs implemented in functional languages, due to both the complex translations of some high level constructs, and the use of automatic memory managers.To help understand memory allocation behavior of Scheme programs, we have designed two complementary tools. The first one reports on frequency of ...

24 Portable, unobtrusive garbage collection for multiprocessor systems

Damien Doligez, Georges Gonthier

February 1994 **Proceedings of the 21st ACM SIGPLAN-SIGACT symposium on Principles of programming languages**

Publisher: ACM Press

Full text available: pdf(1.37 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe and prove the correctness of a new concurrent mark-and-sweep garbage collection algorithm. This algorithm derives from the classical on-the-fly algorithm from Dijkstra et al. [9]. A distinguishing feature of our algorithm is that it supports multiprocessor environments where the registers of running processes are not readily accessible, without imposing any overhead on the elementary operations of loading a register or reading or initializing a field. Furthermor ...

25 Older-first garbage collection in practice: evaluation in a Java Virtual Machine

Darko Stefanović, Matthew Hertz, Stephen M. Blackburn, Kathryn S. McKinley, J. Eliot B. Moss

June 2002 **ACM SIGPLAN Notices , Proceedings of the 2002 workshop on Memory system performance MSP '02**, Volume 38 Issue 2 supplement

Publisher: ACM Press

Full text available: pdf(1.15 MB) Additional Information: full citation, abstract, references, citings

Until recently, the best performing copying garbage collectors used a generational policy which repeatedly collects the very youngest objects, copies any survivors to an older space, and then infrequently collects the older space. A previous study that used garbage-collection simulation pointed to potential improvements by using an *Older-First* copying garbage collection algorithm. The Older-First algorithm sweeps a fixed-sized window through the heap from older to younger objects, and avo ...

26 Combining card marking with remembered sets: how to save scanning time

Alain Azagury, Elliot K. Kolodner, Erez Petrank, Zvi Yehudai

October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3

Publisher: ACM Press

Full text available: pdf(995.83 KB) Additional Information: full citation, abstract, references, citings, index terms

We consider the combination of card marking with remembered sets for generational garbage collection as suggested by Hosking and Hudson [3]. When more than two

generations are used, a naive implementation may cause excessive and wasteful scanning of the cards and thus increase the collection time. We offer a simple data structure and a corresponding algorithm to keep track of which cards need be scanned for which generation. We then extend these ideas for the Train Algorithm of [4]. Here, the so ...

Keywords: garbage collection, generational garbage collection, the train algorithm

27 A parallel, incremental, mostly concurrent garbage collector for servers

Katherine Barabash, Ori Ben-Yitzhak, Irit Goft, Elliot K. Kolodner, Victor Leikehman, Yoav Ossia, Avi Owshanko, Erez Petrank

November 2005 **ACM Transactions on Programming Languages and Systems (TOPLAS)**, Volume 27 Issue 6

Publisher: ACM Press

Full text available: pdf(683.50 KB) Additional Information: full citation, abstract, references, index terms

Multithreaded applications with multigigabyte heaps running on modern servers provide new challenges for garbage collection (GC). The challenges for "server-oriented" GC include: ensuring short pause times on a multigigabyte heap while minimizing throughput penalty, good scaling on multiprocessor hardware, and keeping the number of expensive multicycle fence instructions required by weak ordering to a minimum.We designed and implemented a collector facing these demands building on th ...

Keywords: Garbage collection, JVM, concurrent garbage collection

28 A generational mostly-concurrent garbage collector

Tony Printezis, David Detlefs

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(1.67 MB) Additional Information: full citation, abstract, citings, index terms

This paper reports our experiences with a mostly-concurrent incremental garbage collector, implemented in the context of a high performance virtual machine for the Java™ programming language. The garbage collector is based on the "mostly parallel" collection algorithm of Boehm *et al.* and can be used as the old generation of a generational memory system. It overloads efficient write-barrier code already generated to support generational garbage collection to also ident ...

29 Garbarge collection for Prolog based on WAM

K. Appleby, M. Carllson, S. Haridi, D. Sawhlin

June 1988 **Communications of the ACM**, Volume 31 Issue 6

Publisher: ACM Press

Full text available: pdf(1.91 MB) Additional Information: full citation, abstract, references, citings, index terms

The Warren abstract machine (WAM) has become a generally accepted standard Prolog implementation technique. Garbage collection is an important aspect in the implementation of any Prolog system. A synopsis of the WAM is presented and then marking and compaction algorithms are shown that take advantage of WAM's unique use of the data areas. Marking and compaction are performed on both the heap and the trail; both use pointer reversal techniques, which obviate the need for extra stack space. H ...

30 A principled approach to operating system construction in Haskell

Thomas Hallgren, Mark P. Jones, Rebekah Leslie, Andrew Tolmach

September 2005 **ACM SIGPLAN Notices , Proceedings of the tenth ACM SIGPLAN**

international conference on Functional programming ICFP '05, Volume 40 Issue 9
Publisher: ACM Press
Full text available: pdf(154.82 KB) Additional Information: full citation, abstract, references, index terms

We describe a monadic interface to low-level hardware features that is a suitable basis for building operating systems in Haskell. The interface includes primitives for controlling memory management hardware, user-mode process execution, and low-level device I/O. The interface enforces memory safety in nearly all circumstances. Its behavior is specified in part by formal assertions written in a programming logic called P-Logic. The interface has been implemented on bare IA32 hardware using the G ...

Keywords: Haskell, hardware interface, monads, operating systems, programming logic, verification

31 A "card-marking" scheme for controlling intergenerational references in generation-based garbage collection on stock hardware
P. R. Wilson, T. G. Moher
May 1989 **ACM SIGPLAN Notices**, Volume 24 Issue 5
Publisher: ACM Press
Full text available: pdf(505.66 KB) Additional Information: full citation, citings, index terms

32 Very concurrent mark-&-sweep garbage collection without fine-grain synchronization
Lorenz Huelsbergen, Phil Winterbottom
October 1998 **ACM SIGPLAN Notices , Proceedings of the 1st international symposium on Memory management ISMM '98**, Volume 34 Issue 3
Publisher: ACM Press
Full text available: pdf(1.36 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a new incremental algorithm for the concurrent reclamation of a program's allocated, yet unreachable, data. Our algorithm is a variant of mark-&-sweep collection that---unlike prior designs---runs mutator, marker, and sweeper threads concurrently *without* explicit fine-grain synchronization on shared-memory multiprocessors. A global, but infrequent, synchronization coordinates the per-object coloring marks used by the three threads; fine-grain synchronization is achieve ...

33 Programming languages: Garbage collection for embedded systems
David F. Bacon, Perry Cheng, David Grove
September 2004 **Proceedings of the 4th ACM international conference on Embedded software EMSOFT '04**
Publisher: ACM Press
Full text available: pdf(199.59 KB) Additional Information: full citation, abstract, references, citings, index terms

Security concerns on embedded devices like cellular phones make Java an extremely attractive technology for providing third-party and user-downloadable functionality. However, garbage collectors have typically required several times the maximum live data set size (which is the minimum possible heap size) in order to run well. In addition, the size of the virtual machine (ROM) image and the size of the collector's data structures (metadata) have not been a concern for server- or workstation-orien ...

Keywords: compaction, fragmentation, mark-and-sweep, tracing

34 The case for profile-directed selection of garbage collectors

Robert Fitzgerald, David Tarditi

October 2000 **ACM SIGPLAN Notices , Proceedings of the 2nd international symposium on Memory management ISMM '00**, Volume 36 Issue 1

Publisher: ACM Press

Full text available: pdf(1.28 MB) Additional Information: full citation, abstract, citings, index terms

Many garbage-collected systems use a single garbage collection algorithm across all applications. It has long been known that this can produce poor performance on applications for which that collector is not well suited. In some systems, such as those that execute stand-alone compiled executables, an appropriate collector for each application can be selected from a pool of available collectors and tuned by using profile information. In a study of 20 benchmarks and several collectors, compiled ...

35 Concurrent compacting garbage collection of a persistent heap

James O'Toole, Scott Nettles, David Gifford

December 1993 **ACM SIGOPS Operating Systems Review , Proceedings of the fourteenth ACM symposium on Operating systems principles SOSP '93**, Volume 27 Issue 5

Publisher: ACM Press

Full text available: pdf(1.50 MB) Additional Information: full citation, abstract, references, citings, index terms

We describe a replicating garbage collector for a persistent heap. The garbage collector cooperates with a transaction manager to provide safe and efficient transactional storage management. Clients read and write the heap in primary memory and can commit or abort their write operations. When write operations are committed they are preserved in stable storage and survive system failures. Clients can freely access the heap during garbage collection because the collector concurrently builds a comp ...

36 An on-the-fly mark and sweep garbage collector based on sliding views

Hezi Azatchi, Yossi Levanoni, Harel Paz, Erez Petrank

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11

Publisher: ACM Press

Full text available: pdf(244.12 KB) Additional Information: full citation, abstract, references, citings, index terms

With concurrent and garbage collected languages like Java and C# becoming popular, the need for a suitable non-intrusive, efficient, and concurrent multiprocessor garbage collector has become acute. We propose a novel mark and sweep on-the-fly algorithm based on the sliding views mechanism of Levanoni and Petrank. We have implemented our collector on the Jikes Java Virtual Machine running on a Netfinity multiprocessor and compared it to the concurrent algorithm and to the stop-the-world collecto ...

Keywords: concurrent garbage collection, garbage collection, memory management, on-the-fly garbage collection, runtime systems

37 Segment order preserving copying garbage collection for WAM based Prolog

Bart Demoen, Geert Engels, Paul Tarau

February 1996 **Proceedings of the 1996 ACM symposium on Applied Computing**

Publisher: ACM Press

Full text available: pdf(843.94 KB) Additional Information: full citation, references, citings, index terms

Keywords: WAM based Prolog implementation, memory management of logic programming languages, segment order preserving copying garbage collection

38 Design of the opportunistic garbage collector

P. R. Wilson, T. G. Moher

September 1989 **ACM SIGPLAN Notices , Conference proceedings on Object-oriented programming systems, languages and applications OOPSLA '89**, Volume 24 Issue 10

Publisher: ACM Press

Full text available: pdf(1.33 MB) Additional Information: full citation, abstract, references, citings, index terms

The Opportunistic Garbage Collector (OGC) is a generational garbage collector for stock hardware and operating systems. While incorporating important features of previous systems, the OGC includes several innovations. A new bucket brigade heap organization supports advancement thresholds between one and two scavenges, using only two or three spaces per generation, and without requiring per-object counts. Opportunistic scavenging decouples scavenging from th ...

39 GCspy: an adaptable heap visualisation framework

Tony Printezis, Richard Jones

November 2002 **ACM SIGPLAN Notices , Proceedings of the 17th ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications OOPSLA '02**, Volume 37 Issue 11

Publisher: ACM Press

Full text available: pdf(215.66 KB) Additional Information: full citation, abstract, references, citings, index terms

GCspy is an architectural framework for the collection, transmission, storage and replay of memory management behaviour. It makes new contributions to the understanding of the dynamic memory behaviour of programming languages (and especially object-oriented languages that make heavy demands on the performance of memory managers). GCspy's architecture allows easy incorporation into *any* memory management system: it is not limited to garbage-collected languages. It requires only small change ...

Keywords: Java, garbage collection, language implementation, memory management, visualisation of objects

40 Ulterior reference counting: fast garbage collection without a long wait

Stephen M. Blackburn, Kathryn S. McKinley

October 2003 **ACM SIGPLAN Notices , Proceedings of the 18th annual ACM SIGPLAN conference on Object-oriented programing, systems, languages, and applications OOPSLA '03**, Volume 38 Issue 11

Publisher: ACM Press

Full text available: pdf(218.61 KB) Additional Information: full citation, abstract, references, citings, index terms

General purpose garbage collectors have yet to combine short pause times with high throughput. For example, generational collectors can achieve high throughput. They have modest average pause times, but occasionally collect the whole heap and consequently incur long pauses. At the other extreme, concurrent collectors, including reference counting, attain short pause times but with significant performance penalties. This paper introduces a new hybrid collector that combines copying generational c ...

Keywords: Java, copying, generational hybrid, reference counting, ulterior reference

counting

Results 21 - 40 of 200 Result page: previous 1 **2** 3 4 5 6 7 8 9 10 next